CLAIMS

What is claimed is:

1. A versioning Application Programming Interface (API) for a software platform based on an object-oriented platform-independent programming language, said versioning API comprising:

main interfaces defining versioning functionality, said main interfaces allowing access to the versioning functionality;

a functional implementation of said main interfaces, said functional implementation comprising classes and libraries implementing the versioning functionality, said classes including a reference to a program module to perform a requested versioning function; and

a user interface for using the versioning functionality.

2. A versioning API according to claim 1, further comprising a communication mechanism implementing client-server functionality.

3. A versioning API according to claim 1 wherein said main interfaces comprising:

an interface defining versioning server functionality;

an interface defining versioning client functionality;

an interface defining versioning repository functionality;

an interface defining designated directory structures and access to the designated directory structures; and

an interface defining transactions between the designated directory structures.

4. A versioning API according to claim 3 wherein said main interfaces further comprising:

an interface defining file actions within a designated directory structure.

5. A versioning API according to claim 1, further comprising:

native programming interfaces allowing code written in the object-oriented platform-independent language to operate with code written in a native language other than the object-oriented platform-independent language.

6. A versioning API according to claim 5 wherein said functional implementation comprising:

classes and first libraries written in an object-oriented platform-independent programming language; and

second libraries including software routines written in a native programming language other than the object-oriented platform-independent language, said second libraries implementing said native programming interfaces.

7. A versioning API according to claim 6 wherein said classes comprise:

an implementation class including:

a reference to a first library, said reference being invoked if a requested versioning function is implemented with the object-oriented platform-independent programming language; and

a reference to a native function and a second library, said reference being invoked, using a native programming interface, if a requested versioning function is implemented with the native programming language.

8. A versioning API according to claim 6 wherein said functional implementation further comprising:

resource files available to said classes and libraries.

9. A versioning API according to claim 1 wherein the classes comprise:

a class BringoverFrom including a reference to a program module for copying master files stored in a first directory structure and thereby creating a set of working files; and

a class BringoverTo including a reference to a program module for storing the set of working files in a second directory structure.

10. A versioning API according to claim 9 wherein the classes further comprise:

a class PutbackFrom including a reference to a program module for copying the working files in the second directory structure and thereby creating a set of updated files; and

a class PutbackTo including a reference to a program module for replacing a corresponding set of the master files in the first directory structure with the set of updated files.

11. A versioning API according to claim 10 wherein the classes further comprise:

a class Conflict including:

a reference to a program module for receiving a request for replacing the master files with a set of updated files, and checking for a previous replacement of the master files with another set of updated files;

a reference to said class PutbackTo;

a reference to said class BringoverFrom; and

a reference to said class BringoverTo.

12. A versioning API according to claim 11 wherein said reference to said class PutbackTo is invoked if there is no previous replacement of the master files.

13. A versioning API according to claim 11 wherein said reference to said class BringoverFrom and said reference to said class BringoverTo are invoked if there is a previous replacement of the master files.

14. A versioning API according to claim 10 wherein the classes further comprise:

a class Checkout including:

a reference to a program module for creating a writeable copy of a working file stored in the second directory structure; and

a reference to a program module for storing the writeable copy to a requested address; and

a class Checkin including:

a reference to a program module for copying the writeable copy of a requested address so as to create an updated working file; and

a reference to a program module for replacing the working file with the updated working file.

15. A versioning API according to claim 14 wherein the classes further comprise:

a class Lock including a reference to a program module for receive a request for creating a writeable copy of a working file and checking whether a writeable copy of the working file has already been created.

16. A versioning API according to claim 9 wherein the classes further comprise:

a class Freezepoint including a reference to a program module for creating freezepoint files for files in a specified directory structure, the freezepoint files storing a specific time stamp and a then current version of the corresponding files.

17. A versioning API according to claim 1 wherein said user interface comprises at least one of:

a graphic user interface; and

a command line interface.

18. A method for using version control functionality via a versioning Application Programming Interface (API) provided in a software platform based on an object-oriented platform-independent programming language, said method comprising:

defining versioning functionality in main interfaces of said versioning API;

implementing the versioning functionality in classes and libraries of said versioning API, the libraries including:

first libraries written in an object-oriented platform-independent programming language, and

second libraries written in a native programming language other than the object-oriented platform-independent language; and

providing native programming interfaces allowing code written in the object-oriented platform-independent language to operate with code written in a native language other than the object-oriented platform-independent language, the second libraries including native programming interface implementation.

19. A method according to claim 18, further comprising:

receiving, from a client, a request for a versioning function;

calling a class implementing the requested versioning function;

invoking a first library from the class, if the requested versioning function is implemented in the first library written in the object-oriented platform-independent program language; and

using a native programming interface from the class, so as to invoke a second library if a requested versioning function is implemented in the second library written in a native language other than the object-oriented platform-independent language.

20. A method according to claim 18 wherein the classes and libraries are mounted with a versioning server application deployed to a hosting server running on the software platform, the software platform being based on the object-oriented platform-independent programming language, said method further comprising:

making a call for a method of a proxy object at the client, the proxy object being associated with a type of versioning transaction;

converting the call for a method to a request of the method;

transmitting the request to the hosting server; and

invoking a servlet at the hosting server to generate a response to the request, the servlet delegating processing of the request to a server object calling a class including the requested method;

invoking, from the class, the method directly if the requested method is implemented in a first library written in the object-oriented platform-independent program language; and

invoking, from the class, the method using a native programming interface if the requested method is implemented in a second library written in a native language other than the object-oriented platform-independent program language.

21. A method according to claim 20 wherein said making a call is performed with a graphic user interface.

22. A method according to claim 20 wherein said making a call is performed with a command line interface.

23. A method according to claim 18 wherein said defining versioning functionality comprising:

defining versioning server functionality;

defining versioning client functionality;

defining versioning repository functionality;

defining designated directory structures access to the designated directory structures; and

defining transactions between the designated directory structures.

24. A method according to claim 23 wherein said defining versioning functionality further comprising:

defining file actions within a designated directory structure.

25. A method according to claim 18 wherein in said implementing, the versioning functionality is further implemented in resource files available to the classes and libraries.

26. A method according to claim 18 wherein the versioning functionality implemented in classes and libraries comprises:

copying master files stored in a first directory structure and thereby creating a set of working files; and

storing the set of working files in a second directory structure.

27. A method according to claim 26 wherein the versioning functionality implemented in classes and libraries further comprises:

copying the working files in the second directory structure and thereby creating a set of updated files; and

replacing the master files in the first directory structure with the set of updated files.

28. A method according to claim 27 wherein the versioning functionality implemented in classes and libraries further comprises:

receiving a request for replacing the master files with a set of updated files, and checking for a previous replacement of the master files with another set of updated files;

calling said replacing if there is no previous replacement of the master files since a previous copying of the master files; and

calling said copying master files and said storing if there is a previous replacement of the master files since a previous copying of the master files.

29. A method according to claim 27 wherein the versioning functionality implemented in classes and libraries further comprises:

creating a writeable copy of a working file stored in the second directory structure; and

storing the writeable copy to a requested address.

30. A method according to claim 29 wherein the versioning functionality implemented in classes and libraries further comprises:

copying the writeable copy so as to create an updated working file; and

replacing the working file in the second directory with the updated working file.

31. A method according to claim 30 wherein the versioning functionality implemented in classes and libraries further comprises:

receive a request for creating a writeable copy of a working file; and

checking whether a writeable copy of the working file has already been created.

32. A method according to claim 27 wherein the versioning functionality implemented in classes and libraries further comprises:

creating freezepoint files for files in a specified directory structure, the freezepoint files storing a specific time stamp and a then current version of the corresponding files.